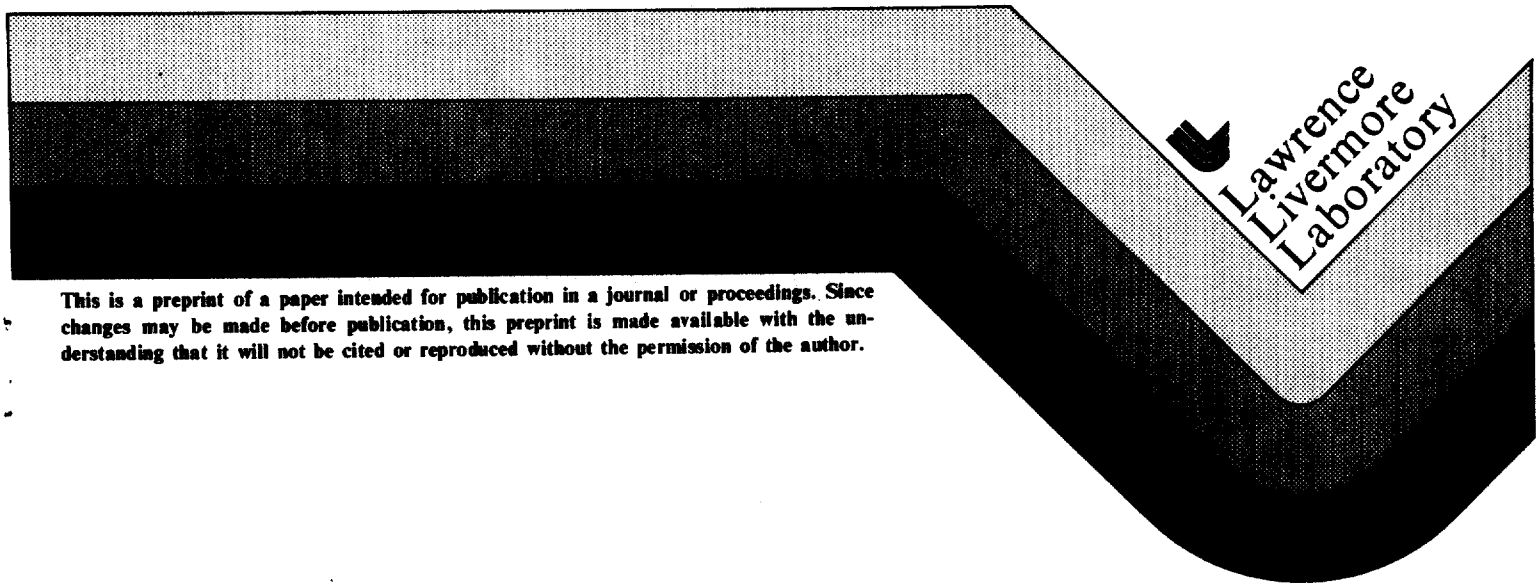
UCRL- 82957
PREPRINT

RESULTS OF STUDIES PERFORMED ON THE MODEL OF THE MFTF SUPERVISORY CONTROL AND DIAGNOSTICS SYSTEM (SCDS) 1-6

ROBERT H. WYMAN

This paper was prepared for submittal to the
8th SYMPOSIUM ON ENGINEERING PROBLEMS OF
FUSION RESEARCH; IEEE; SHERATON HOTEL,
SAN FRANCISCO, CA., NOVEMBER 13-16, 1979

11-12-79



RESULTS OF STUDIES PERFORMED ON THE MODEL OF THE MFTF SUPERVISORY CONTROL AND DIAGNOSTICS SYSTEM (SCDS)[1-6]*

Robert H. Wyman
Lawrence Livermore Laboratory, University of California
Livermore, California 94550

Introduction

The design and implementation of the SCDS is a relatively complex problem involving a nine-computer network coupled with a unique color graphics control console system, 50 local control minicomputers, and the usual array of drives, printers, magnetic tapes, etc. Four million bytes of data are to be collected on each MFTF cycle with a repetition rate of five minutes per shot, and the associated data processing and storing load is a major concern.

Crude paper studies were made initially to try to size the various components of the system and various configurations were proposed and analyzed prior to the solicitation for the computer system. However, once the hardware was purchased and a preliminary software design was completed, it became essential and feasible to do an analysis of the system to considerably greater depth in order to identify bottlenecks and other system problems and to verify those parts of the design that met the MFTF requirements.

The Simulation Language

Although several simulation languages are available at LLL, the one best fitting our needs is ASPOL[7], a process-oriented simulation language based partially on SOL.[8,9] The language has a strong PASCAL[10] flavor with a FORTRAN aftertaste for reasons which will become clear in the section on the ASPOL system.

The language contains several constructs facilitating simulation of systems such as SCDS:

1. Facilities which can be reserved, held for a period of time, and then released. (Examples of Facilities are central processors (CPU) and disks.)

2. Storages which can be given a size and then are allocated and deallocated as required by the simulation. (An example of a storage is the SCDS shared memory.)

3. Processes which can be initiated and which then perform some task in the simulation. (An example of a process in SCDS is TICK, which is initiated once per simulation-second per CPU and updates the clock display on the console.)

4. Events which can be considered process synchronizers. An event can be set by one process and if another process is waiting on that event, the waiting process will then continue and the event will be automatically reset. (An example of an event is SHOOT. At the beginning of an MFTF shot a number of processes are initiated to prepare for the shot. When they complete their functions, they wait for the SHOOT event which signals the firing of the neutral beam system. When SHOOT is set, they then all proceed into the data collection and storage cycle.)

*Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

Additionally, the language has a number of built-in random function generators with various common distributions for easily including statistical variations in model activities. Also, the language has a number of statistics-collecting features which allow easy output in a standard form of distributions occurring within the operation being simulated.

Naturally, the language has the usual computational, logical, input-output, and file-handling features found in most computer languages, but it also has a macro feature which is extremely useful, since subroutines in the language may not make use of many of the simulation statements.

The ASPOL System

The ASPOL system is composed of the following:
1. ASPOL-to-FORTRAN translator.
2. FORTRAN compiler.
3. Error-monitoring and trace subroutines.
4. Input/output system.
5. Library.

In order to minimize the translation effort, many FORTRAN constructs are used in the language. However it is possible to write GOTO-free code in ASPOL. Unfortunately, the object code cannot be saved after compilation, hence, translation and compilation (no small task) must occur for each run if intervention is required between runs.

A number of features are included by the system for handling facilities storages, processes and events. When multiple processes, request a facility, or when more storage is requested than is available, the system will queue the requests and process them in FIFO order. (Traces of the code have indicated that this is violated occasionally.) Further, a priority system is available which allows preemption of facilities with creation of multiple queues for the various priorities, etc. Statistics on these queues, as well as durations of facilities use are collected automatically and listed at termination time.

A process, unlike a subroutine, may have multiple copies running simultaneously, each with its own set of local variables. ("Simultaneously" here means that the simulation clock does not advance during execution.) Every time a process is initiated by another process, a new copy is created, and the system handles all details of keeping the variable sets separate and storing the locations of those instructions which cause a process to temporarily suspend execution. (A process terminates only when it encounters an END statement and it is true termination without a RETURN type operation.) The system will also reinitiate the suspended process when the cause of suspension is removed, e.g. a facility becomes available and can be assigned to the process, an event occurs for which the process was waiting, or the time specified by a HOLD operation elapses.

At LLL, ASPOL runs under the Slope operating system, a version of one of CDC's systems, which in turn runs under the LTSS operating system on the CDC 7600 machines. Typically, the SCDS simulation requires 10 seconds of translation time, 5 seconds

of compilation time, 22 seconds of execution time, and 77 seconds of I/O time for a 15-minute model simulation encompassing two MFTF shots.

The Model

The system being modelled, the SCDS, is shown in Figure 1. The model then contains nine CPUs; eleven disks; seven display channels, one for each console; and 128 kilobytes of shared memory. The shared memory is broken up into one 64-kilobyte block for general communication, one 32-kilobyte block which is used to move the 4 megabytes of plasma diagnostics data, and one 32- kilobyte block which is used for data-base operations.

Disks are assumed to have a seek time given by

$$t = \frac{t_{min}}{\sqrt{d^2 + b^2}} \qquad (d \neq 0) ,$$

where

$$b^2 = \frac{t_{min}^2 \; d_{max}^2}{t_{max}^2 - t_{min}^2} ,$$

d is the number of cylinders over which the heads are to move, t_{min} is the single-track move time, and t_{max} is the time for the maximum distance move. This curve is a hyperbola which intuitively seems to have two characteristics for small head motions, the time is more or less independent of d (i.e., $b^2 >> d^2$) for large head motions, t is proportional to d (i.e., $d^2 >> b^2$). Naturally, for d = 0, t = 0, and this is accounted for in the model.

Also, read and write operations always use the time for an integer number of sectors to move past a fixed point.

Thus, a typical disk operation starts with calculation of a head traverse d by taking the absolute value of the difference between two random integers each chosen from a uniform distribution between 1 and the maximum track number for the particular disk. This difference is then used to calculate the seek time using the formula given above. To this is added a latency time chosen from a uniform distribution over the range zero to one disk rotation time, and then to this sum is added a random integer multiple of sector times. The disk is then reserved, held for this time, and released.

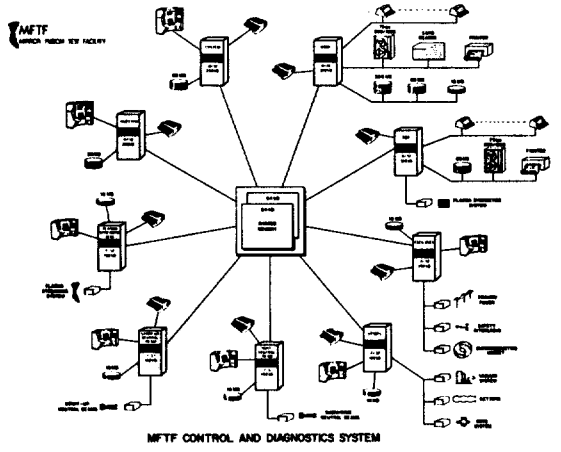


Figure 1

For a directory search, an exponentially distributed number of these operations are performed with two modifications. The latency time after the first seek is assumed to be one revolution minus some small processing time, and the number of sectors read is always one. The distribution has an average value of two in the current model, and, naturally, the number of operations is an integer.

Three primary processes control most of the model activity:

- The control panel process
- The display updater process
- The shot process

The control panel process, seven copies of which run continuously from the start of the simulation until termination, generate control panel "button pushes." That is, at random times with a normal distribution it is assumed that an operation initiates a system action. These actions are classified as

- Bring up a new control panel (40%).
- Bring up a new display (20%).
- Change a system parameter (30%).
- Make a request for historical data from the data-base (10%).

The numbers in parentheses are the relative proportions of the various actions. Each action is performed by a process which uses the various facilities required for times estimated by the programmers of the system.

The display updater process is required, naturally, to simulate the dynamic changing of the data displayed on the many color-graphics displays of the console system. This process is actually a group of processes that do the following:

- Keep track of when a display (or displays) need to be updated. (This is the initiator process.)
- Start the necessary processng on the CPU responsible for a display. (This is the updater process.)
- Initiate the actual updating on each CPU which is currently displaying the particular display being updated, wait for all updating to complete, and then terminate the initiator. (This is the waiter process.)
- Do the necessary processing on the CPUs containing the display being updated. (This is the display update process.)

This whole sequence makes use of two tables; one holds the location of each display currently being displayed, and the other is a table of all available displays which contains the CPU responsible for a display's maintenance, whether it is updated periodically, randomly, or never, the time to next update, and the period if the update is periodic.

The shot process is again a group of processes:

- The shot initiator "awakens" every five minutes, starts all the remaining shot processes, waits for all to complete the arming operations, and then starts the data collection and storage.

- The neutral-beam process opens a local file for temporary storage of the beam data and then waits for the initiator. When the initiator releases it, it collects the beam data, sends them via shared memory to the data base manager and simultaneously writes the data to the local disk. Twenty copies of this process execute, using one CPU to simulate the startup beams, and 24 copies execute on another CPU for the sustaining system.

- The plasma-gun process is similar to the neutral-beam process, but it has considerably less data to handle, and only one copy executes.

- The diagnostics system process is also similar, except that it runs on its own CPU, it exists in only one copy, and it moves blocks of data through shared memory to the data base manager. Four million bytes are transferred, by far the biggest block of data.

- The data base manager process receives all data from the above processes and writes it onto the big disk of the system.

As might be expected, none of the processes described above is as clean and "pure" as indicated, since a number of complications and special cases must be handled, and a number of subsidiary issues add to the complexity. Nevertheless, the main framework of the system is as described.

The Results

The results of the studies performed to date have been credible and useful. There are always doubts about how well a model like this reflects reality, but conversations with various members of the SCDS group about execution times of various pieces of code seem to indicate that, if our model is wrong, at least it is failing in the direction which is safest, i.e., if the model indicates no performance problems in a specific area, then real life has a very high probability of not having a problem in the same area, whereas if the model does have a problem in another area, there is a fair chance that real life may not have a problem in that area.

The model was run initially with only operators pushing control panel buttons, no display updating, and no shot-data being collected. This pointed up some problems with the execution time specified for a particular operation. When this problem was corrected, it was immediately clear that operators pushing buttons posed almost no load to the system. A response-time table was prepared that records the time from button push to operation completion: over all model runs, the distribution of response times has changed very little. A typical response time table is shown in Fig. 2, which was generated from a model that did not have shot-data processing included. All times shown are in milliseconds.

ASPOL DATA REPORT

TIME 999999.99

TABLE - RESPONSE TIME
TABLE NO. 44

MINIMUM 66.960 MEAN 512.859
MAXIMUM 1469.481 STD. DEV. 378.936
NUMBER OF ENTRIES 94

UPPER LIMIT	FREQUENCY	PROPORTION	CUMULATIVE PROPORTION
100.000	12	.1277	.1277
200.000	16	.1702	.2979
300.000	7	.0745	.3723
400.000	8	.0532	.4255
500.000	14	.1064	.5319
600.000	15	.1064	.6383
700.000	9	.0957	.7340
800.000	4	.0426	.7766
900.000	5	.0532	.8298
1000.000	5	.0532	.8830
1100.000	3	.0319	.9149
1200.000	3	.0319	.9468
1300.000	1	.0106	.9574
1400.000	2	.0213	.9787
1900.000	2	.0213	1.0000

Figure 2

The next item added to the model was display updating. A very crude algorithm was implemented first. This was to update every display once per second and to simply replace the entire display, rather than any the changed information. This scheme choked the shared memory to the point where the update rate could not be met. Some internal queues began overflowing to indicate the problem.

The update algorithm was then modified to allow three kinds of updating:

- Periodic
- Random
- None

Further, the amount of data required for updating a display was reduced to 25% of the original display size.

The display update algorithm also allows an unequal distribution of display activity, i.e., one machine may be responsible for many more displays than another, and, independently, a particular display may appear in 0, 1 or many places. (One hundred displays and 27 color CRTs are available in the model. A display which does not appear on a CRT is never included in the update algorithm. However, a new display may be selected by an operator to replace an old display on a particular CRT and all of this activity is included in the model.)

After this rather elaborate modification to the update algorithm, statistics were collected on the update time (which is measured from the time the responsible CPU starts the update until the last CPU where the display appears has responded and the responsible CPU terminates). A table of update times is given in Fig. 3. Here again, no shot-data processing is included.

ASPOL DATA REPORT

TIME 999999.99

TABLE - TIME TO UPDATE
TABLE NO. 42

MINIMUM 544.264 MEAN 979.268
MAXIMUM 2312.648 STD. DEV. 242.782
NUMBER OF ENTRIES 4548

UPPER LIMIT	FREQUENCY	PROPORTION	CUMULATIVE PROPORTION
100.000	0	0.0000	0.0000
200.000	0	0.0000	0.0000
300.000	0	0.0000	0.0000
400.000	0	0.0000	0.0000
500.000	0	0.0000	0.0000
600.000	21	.0046	.0046
700.000	237	.0500	.0546
800.000	1046	.1907	.2454
900.000	1009	.2283	.4686
1000.000	744	.1539	.6298
1100.000	540	.1109	.7408
1200.000	383	.0778	.8262
1300.000	206	.0430	.8892
1400.000	204	.0449	.9341
1500.000	128	.0282	.9623
1600.000	76	.0164	.9778
1700.000	39	.0098	.9863
1800.000	20	.0052	.9926
1900.000	17	.0037	.9963
2000.000	5	.0020	.9982
2100.000	5	.0011	.9993
2200.000	2	.0004	.9998
2300.000	0	0.0000	.9998
2400.000	1	.0002	1.0000

Figure 3

The inclusion of shot-data processing produced some very interesting problems. Initially, the entire shared memory was made available for handling shot-data. Three facts emerged from this:

- Shared memory filled up for several tens of seconds after shot time.
- The maximum time for display update changed from 2312.64 milliseconds to 12118.8 milliseconds. A dramatic increase.
- The maximum response time changed from 1489.48 milliseconds to 2187.8 milliseconds.

The values for the second and third items above are given in Figs. 2-5. Note the long tail which develops in update time (Figs. 3 and 5) as shot-data processing is included in the model. A similar phenomenon develops in response time (Figs. 2 and 4), but it is not so pronounced.

A S P O L D A T A R E P O R T

TIME 888888.88

TABLE - RESPONSE TIME
TABLE NO. 44

MINIMUM 55.968
MAXIMUM 2187.883

MEAN 565.899
STD. DEV. 397.539

NUMBER OF ENTRIES 91

UPPER LIMIT	FREQUENCY	PROPORTION	CUMULATIVE PROPORTION
100.000	18	.1899	.1899
200.000	11	.1209	.2308
300.000	3	.0330	.2637
400.000	6	.0659	.3297
500.000	15	.1648	.4945
600.000	6	.0659	.5604
700.000	11	.1209	.6813
800.000	8	.0989	.7802
900.000	6	.0659	.8461
1000.000	2	.0220	.8681
1100.000	2	.0220	.9121
1200.000	2	.0220	.9341
1300.000	2	.0220	.9560
1400.000	1	.0110	.9670
1500.000	0	0.0000	.9670
1600.000	1	.0110	.9780
1700.000	0	0.0000	.9780
1800.000	1	.0110	.9890
1900.000	0	0.0000	.9890
2000.000	0	0.0000	.9890
GREATER THAN 2000.000	1	.0110	1.0000

Figure 4

A S P O L D A T A R E P O R T

TIME 888888.88

TABLE - TIME TO UPDATE
TABLE NO. 42

MINIMUM 593.162
MAXIMUM 12118.888

MEAN 1121.598
STD. DEV. 894.420

NUMBER OF ENTRIES 3330

UPPER LIMIT	FREQUENCY	PROPORTION	CUMULATIVE PROPORTION
100.000	0	0.0000	0.0000
200.000	0	0.0000	0.0000
300.000	0	0.0000	0.0000
400.000	0	0.0000	0.0000
500.000	0	0.0000	0.0000
600.000	1	.0003	.0003
700.000	98	.0294	.0297
800.000	489	.1468	.1762
900.000	693	.2078	.3838
1000.000	512	.1534	.5371
1100.000	378	.1132	.6504
1200.000	298	.0893	.7397
1300.000	262	.0786	.8182
1400.000	172	.0515	.8697
1500.000	137	.0410	.9107
1600.000	73	.0219	.9326
1700.000	53	.0159	.9485
1800.000	30	.0090	.9575
1900.000	27	.0081	.9656
2000.000	17	.0051	.9707
2100.000	13	.0039	.9746
2200.000	14	.0042	.9787
2300.000	8	.0024	.9811
2400.000	10	.0030	.9841
2500.000	2	.0006	.9847
2600.000	3	.0009	.9856
2700.000	2	.0006	.9862
2800.000	2	.0006	.9868
2900.000	1	.0003	.9871
3000.000	3	.0009	.9880
GREATER THAN 3000.000	39	.0120	1.0000

Figure 5

Processing of shot-data (i.e., collection, transmission to the data base manager while simultaneously recording on the local disk, and the subsequent recording on the big disk of the data base manager) was timed as requiring 75 seconds. When shared memory was doubled in size, it still filled up and 75 seconds was still required for processing the shot-data; from this it was concluded that the big disk--and not the shared memory--was the bottleneck.

Changing the model slightly so that part of shared memory was devoted to handling shot-data while part was devoted to other types of interprocessor communication (operator actions and display updating) did not markedly improve the situation. Secondly, it was concluded that the long tail on display updating during shot-data processing was due to busy-ness on the part of processors involved in shot-data processing; hence, they did not respond well when a display update occurred while processing was ongoing.

Figures 6 and 7, are included to show facility utilization. Figure 6 does not include shot-data processing, while Fig. 7 does. In both cases, CPU 3 is busy about 60% of the time. This is attributed to the large display update responsibility of CPU 3, which has no shot-data responsibility. Figure 8 gives the duties of each of the CPUs.

A S P O L D A T A R E P O R T

TIME 888888.88

ACTUAL RUN TIME = 19.416 SECONDS
ACTUAL CM USED = 8745888 WORDS

FACILITY STATISTICS

FACILITY	UTILIZATION	MEAN BUSY PERIOD	NUMBER OF REQUESTS	NUMBER OF INTERRUPTS
CPU 1	.84	4.829	9427	
CPU 2	.88	4.883	17268	
CPU 3	.61	18.382	29989	
CPU 4	.51	28.358	22689	
CPU 5	.37	16.797	19998	
CPU 6	.39	19.942	17581	
CPU 7	.15	12.226	10929	
CPU 8	.88	7.881	85	
CPU 9	.88	6.882	229	
DISK 1	.11	22.628	4523	
DISK 2	.22	22.698	8696	
DISK 3	.58	31.958	16581	
DISK 4	.44	32.574	12282	
DISK 5	.36	31.218	10492	
DISK 6	.34	32.735	9362	
DISK 7	.18	29.663	5387	
DISK 8	.88	23.598	58	
DISK 9	.88	28.878	82	
DISK 10	0.88	0.888	8	
DISK 11	.88	28.493	83	
DISPCHN 1	.82	18.851	1891	
DISPCHN 2	.84	14.888	2884	
DISPCHN 3	.84	21.122	1889	
DISPCHN 4	.81	7.888	1888	
DISPCHN 5	.84	21.245	1983	
DISPCHN 6	.81	9.495	1178	
DISPCHN 7	.85	21.998	1988	

STORAGE STATISTICS

STORAGE	UTILIZATION	OCCUPANCY MEAN	OCCUPANCY MAX.	NUMBER OF REQUESTS
SHARMEM	.85	3.83	9	4624
DIAGMEM	0.88	0.88	8	8
BEAMMEM	0.88	0.88	8	8

Figure 6

A S P O L D A T A R E P O R T

TIME 888888.88

ACTUAL RUN TIME = 18.491 SECONDS
ACTUAL CM USED = 8765888 WORDS

FACILITY STATISTICS

FACILITY	UTILIZATION	MEAN BUSY PERIOD	NUMBER OF REQUESTS	NUMBER OF INTERRUPTS
CPU 1	.83	4.698	6645	21
CPU 2	.85	4.816	11179	
CPU 3	.63	17.521	32273	
CPU 4	.16	10.152	7876	
CPU 5	.42	18.874	19327	883
CPU 6	.46	20.448	19388	633
CPU 7	.17	12.868	11843	3
CPU 8	.83	11.588	2222	26
CPU 9	.87	11.649	6218	166
DISK 1	.87	22.437	2782	
DISK 2	.14	22.684	5473	
DISK 3	.58	31.669	16487	
DISK 4	.14	32.568	3918	
DISK 5	.33	31.728	9244	
DISK 6	.35	32.981	9637	
DISK 7	.19	29.885	5758	
DISK 8	.83	48.728	428	
DISK 9	.82	38.368	61	
DISK 10	0.88	0.888	8	
DISK 11	.85	36.551	1266	
DISPCHN 1	.81	8.185	1481	
DISPCHN 2	.83	11.875	2898	
DISPCHN 3	.86	24.782	2333	
DISPCHN 4	.81	8.648	1838	
DISPCHN 5	.83	17.328	1632	
DISPCHN 6	.81	10.424	1169	
DISPCHN 7	.86	21.987	1971	

STORAGE STATISTICS

STORAGE	UTILIZATION	OCCUPANCY MEAN	OCCUPANCY MAX.	NUMBER OF REQUESTS
SHARMEM	.88	18.48	98	4214
DIAGMEM	.18	.28	2	497
BEAMMEM	.11	1.83	14	786

Figure 7

```
**********************************************************************
**********************  FACILITIES SECTION  **************************
**********************************************************************


COMMENT  THERE ARE NINE CPU'S IN THIS SYSTEM.

           CPU1  IS THE SYSTEM SUPERVISOR
           CPU2  IS THE INJECTOR SUPERVISOR
           CPU3  IS THE FACILITIES SUPERVISOR
           CPU4  IS THE VESSEL SUPERVISOR
           CPU5
           CPU6  ARE THE BEAM SUPERVISORS
           CPU7
           CPU8  IS THE DIAGNOSTICS DATA PROCESSOR
           CPU9  IS THE DATA BASE MANAGER;

FACILITY   CPU(9);

COMMENT  THERE ARE ELEVEN DISKS, ONE ON EACH CPU PLUS TWO ADDITIONAL
         DISKS ON CPU9.

           DISK1  IS A 67 MEGABYTE DISK
           DISK2  SAME AS 1
           DISK3

           THRU   ARE ALL 1# MEGABYTE DISKS

           DISK7
           DISK8  IS A 67 MEGABYTE DISK
           DISK9  IS A 1# MEGABYTE DISK
           DISK1# IS A 67 MEGABYTE DISK
           DISK11 IS A 3## MEGABYTE DISK;

FACILITY   DISK(11);
```

Figure 8

Conclusion

To date, the model has been very helpful. Because of the language, there was not much effort was involved in putting the model together, and results so far have been credible and indicate that there are no severe problems in the system. It is anticipated that studies will be continued and the model refined as the design of SCDS develops further.

References

1. D. Butner, MFTF Supervisory Control and Diagnostics System Hardware, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

2. W. Ng, MFTF Supervisory Control and Diagnostics System Software, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

3. J. H. Choy and J. A. Wade, A Data Base Management System for the MFTF, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

4. J. A. Wade and J. H. Choy Control and Diagnostics Data Structures for the MFTF, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

5. D. M. Nowell and G. D. Bridgeman, MFTF Exception Handling, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

6. P. R. McGoldrick, SCDS Distributed Operating System, Proceedings of the 8th Symposium on Engineering Problems of Fusion Research, November 1979.

7. ASPOL Reference Manual, Control Data Corporation Professional Services Division, Publication No. 17314200, October 1976.

8. D. E. Knuth and J. L. McNeley, SOL-A Symbolic Language for General Purpose Simulation, IEEE Transactions EC 13, August 1964.

9. D. E. Knuth and J. L. McNeley, SOL-A Formal Definition of SOL, IEEE Transactions EC 13, August 1964.

10. Kathleen Jensen and Niklaus Wirth, PASCAL User Manual and Report, 2nd Edition, Springer-Verlag, 1974.